FM3

MOTOR GRAPHICAL INTERFACE

SPANSION

32-BIT MICROCONTROLLER
FM3 Family

USER MANUAL

Publication Number AN706-0006301v1-E Revision 1.0 Issue Date June 26, 2014

Table of Contents

1. Introduction 6
   - 1.1 Purpose 6
   - 1.2 Definitions, Acronyms and Abbreviations 6
   - 1.3 Document Overview 6
2. System Scope 7
   - 2.1 System Structure 7
   - 2.2 System Runtime Environment 8
3. MGI and Features 9
   - 3.1 System Structure 9
     - 3.1.1 Home Page 9
     - 3.1.2 Menu 10
   - 3.2 New/Edit Project Page 10
     - 3.2.1 New Project 10
     - 3.2.2 Edit Project 11
   - 3.3 Add/Edit Product Page 11
     - 3.3.1 Add Product 11
     - 3.3.2 Edit Product 12
   - 3.4 Edit Product Motors 12
   - 3.5 Work Page 13
     - 3.5.1 System Tab 14
     - 3.5.2 Motor Tab 15
     - 3.5.3 Observation Tab 18
     - 3.5.4 Import Tab 19
4. Use Example 21
   - 4.1 Driver Installation 21
   - 4.2 New Project 21
   - 4.3 Edit Project 22
   - 4.4 Edit/Add Product 24
     - 4.4.1 Add Product 24
     - 4.4.2 Edit Product 26
   - 4.5 Connect/Read/Write 27
   - 4.6 Observation 28
   - 4.7 Export Data and Import Data 29
5. Customer Control Interface in FW 31
   - 5.1 UART Selection 31
   - 5.2 Enable/Disable Switch 31
   - 5.3 Dotting Frequency Control for Waveform and Value Display 31
   - 5.4 Observation of User-Defined Signals 32
6. Transplanting MGI to Your Application 33
   - 6.1 Necessary Files 33
   - 6.2 Configure UART to Meet Your Hardware 33
   - 6.3 Add Code to Your Application 34
7. Troubleshooting 37
   - 7.1 Connection Error 37
   - 7.2 Operation Error 37
   - 7.3 Operation Warning 37
8. Additional Information 38
9. Reference Documents 39

Figures

Figure 2-1: System Structure 7
Figure 2-2: Demo System 7
Figure 3-1: Home Page 9
Figure 3-2: Menu 10
Figure 3-3: New Project 10
Figure 3-4: Add Product 11
Figure 3-5: Edit Product Motors 12
Figure 3-6: Work Page 13
Figure 3-7: System Tab 14
Figure 3-8: Motor Tab 15
Figure 3-9: Signals Defined by Users 17
Figure 3-10: Definition of User Interface for ‘Any signals’ 17
Figure 3-11: Assign Signal to the ‘Any signal 1’ 18
Figure 3-12: Observation Tab 18
Figure 3-13: Comparison Tab 19
Figure 4-1: New Project Step1 21
Figure 4-2: New Project Step2 22
Figure 4-3: Edit Project Step1 22
Figure 4-4: Edit Project Step2 23
Figure 4-5: Add Product Step2 24
Figure 4-6: Add Product Step3 25
Figure 4-7: Add Product Step4 25
Figure 4-8: Edit Product Step2 26
Figure 4-9: Edit Product Step4 26
Figure 4-10: Connect/Read/Write Step2 27
Figure 4-11: Connect/Read/Write Step3 28
Figure 4-12: Observation Step3 29
Figure 4-13: Export Data 29
Figure 4-14: Import Data 30
Figure 6-1: Necessary Files to Support MGI 33
Figure 6-2: MCU UART Initialization 34
Figure 6-3: Configuring UART Communication Variables 34
Figure 6-4: Adding the Functions of Receiving Data and Analyzing Data 35
Figure 6-5: Adding the Functions of Data Recorder 35
Figure 6-6: Adding the Interrupt Functions 36

Tables

Table 2-1: System Runtime Environment 8
Table 3-1: Home Page Components 9
Table 3-2: New Project Page Components 11
Table 3-3: Product Page Components 12
Table 3-4: Product Motors Page Components 13
Table 3-5: System Tab Components 14
Table 3-6: Motor Tab Components 16
Table 3-7: Observation Tab Components 19
Table 3-8: Comparison Tab Components 20
Table 5-1: UART Selection 31
Table 5-2: Enable/Disable Switch 31
Table 5-3: Dotting Frequency Control 31
Table 5-4: Observation of User-Defined Signals 32

1. Introduction

1.1 Purpose

This document describes how to use Motor Graphical Interface (MGI) to control and interact with the motor control platform. This software can send/read data to/from motor control platform, and send commands to get continuous data of parameters.

1.2 Definitions, Acronyms and Abbreviations

GUI	Graphics User Interface
MGI	Motor Graphical Interface
HW	Hardware, in this document it means motor control platform hardware board
FW	Firmware

1.3 Document Overview

The rest of document is organized as the following:

Chapter 2 explains *System Scope.*

Chapter 3 explains *MGI and* Features.

Chapter 4 explains *Use* Example.

Chapter 5 explains *Customer Control Interface in* FW.

Chapter 6 explains *Transplanting MGI to Your Application*

Chapter 7 explains *Troubleshooting.*

2. System Scope

2.1 System Structure

The MGI is compatible with 32-bit and 64-bit Windows 7 operating systems installed with Microsoft .Net Framework 4.0. It connects the HW through the FTDI USB- to-UART Convertor.

Figure 2-1: System Structure

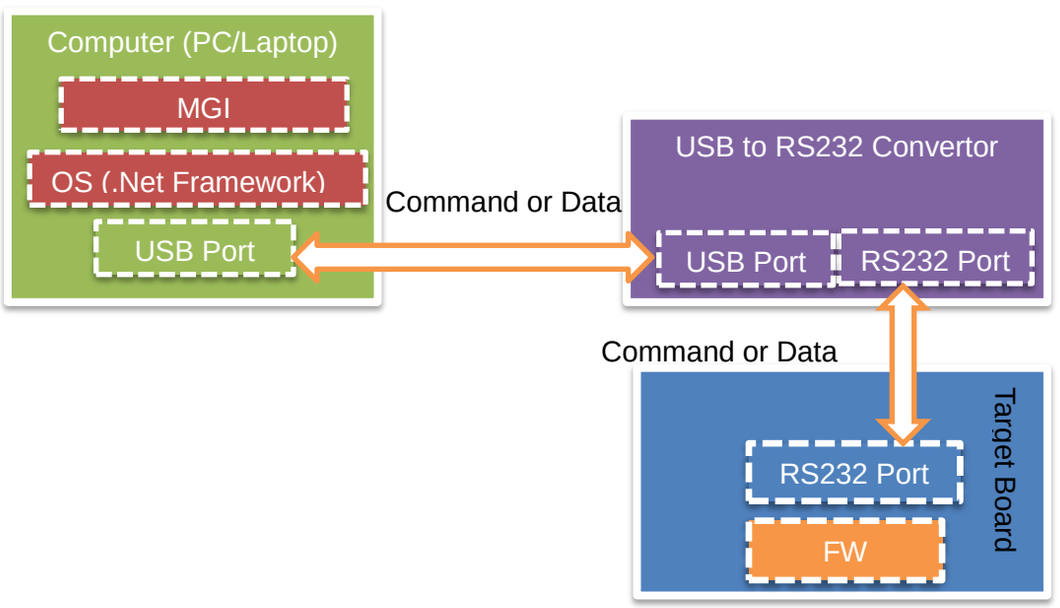
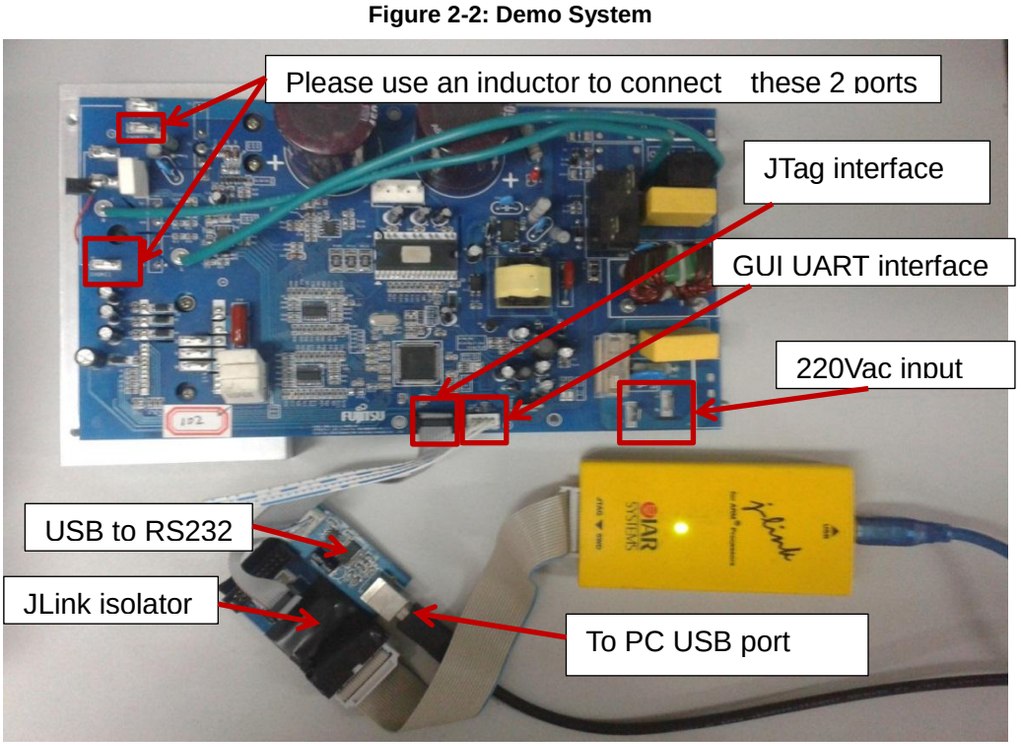


Figure 2-2: Demo System



Notes:

1. *Please confirm the demo board is powered by 220Vac input otherwise the GUI FW cannot work normally because some protection routines hinder GUI FW running.*

2.2 System Runtime Environment

Table 2-1: System Runtime Environment

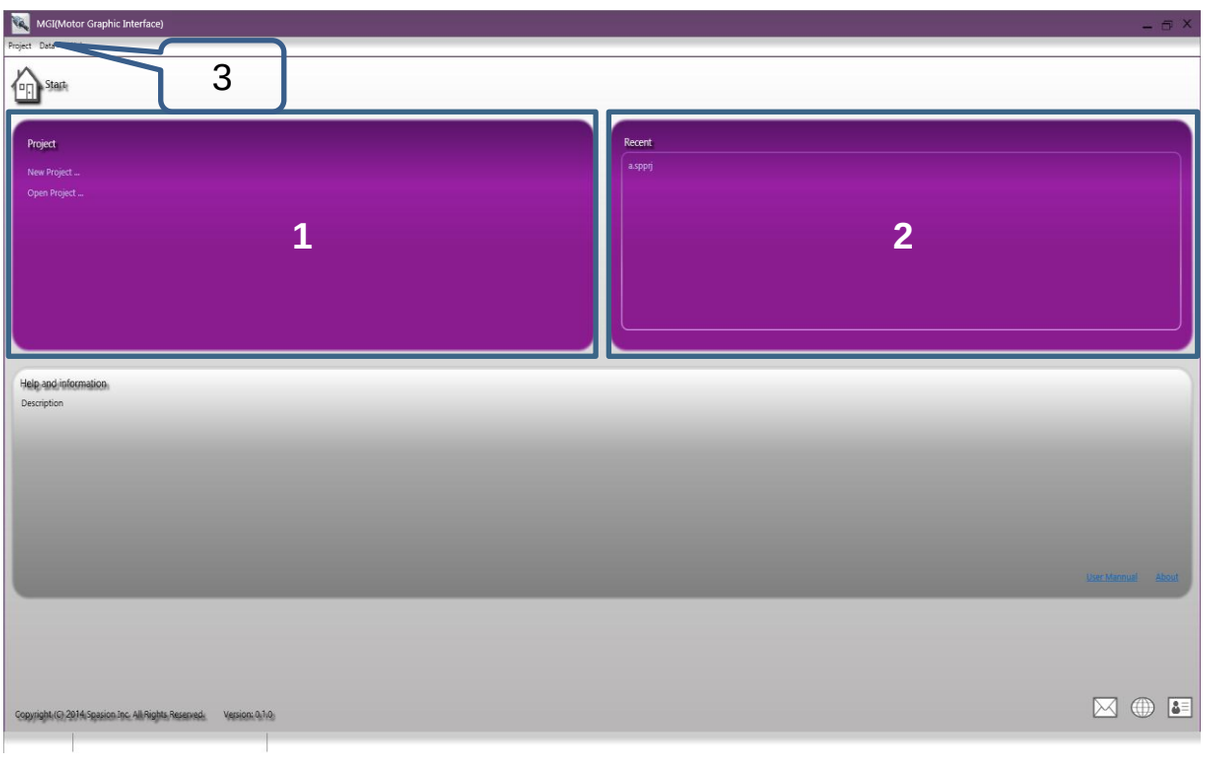
Item	Details
Operating System	Microsoft Windows 7
Runtime Environment	.NET Framework 4.0
IAR EWARM	V7.30 or above (it is necessary only for running demo project)

3. MGI and Features

3.1 System Structure

3.1.1 Home Page

Figure 3-1: Home Page



At the homepage, in the “Project area”, click “New Project” to create a new project and the “New Project” page will be displayed. Then, click “Open Project “to select a project file from the file picker dialog box.

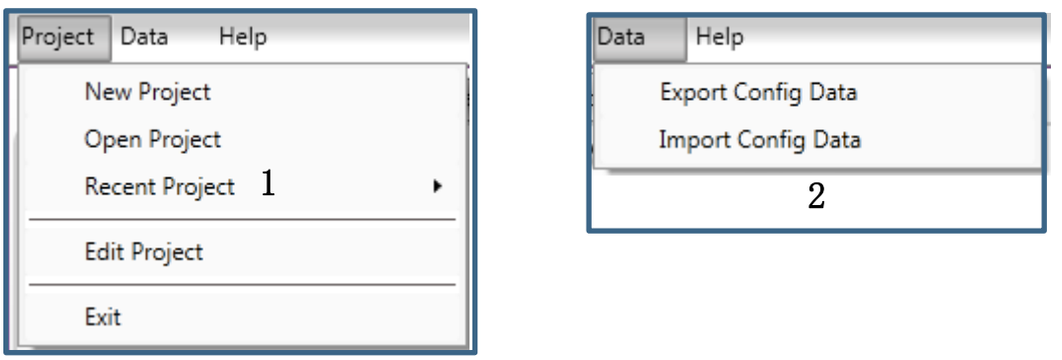
At “Recent” area, the projects that are used before will be listed if the project exists on this PC/Laptop. Click the project name to open the project.

Table 3-1: Home Page Components

Number	Name	Functions
1	Project Area	1. New Project: Create new project. 2. Open Project: Open an exist project.
2	Recent Area	Recent Projects: Open a project from recent project

3.1.2 Menu

Figure 3-2: Menu

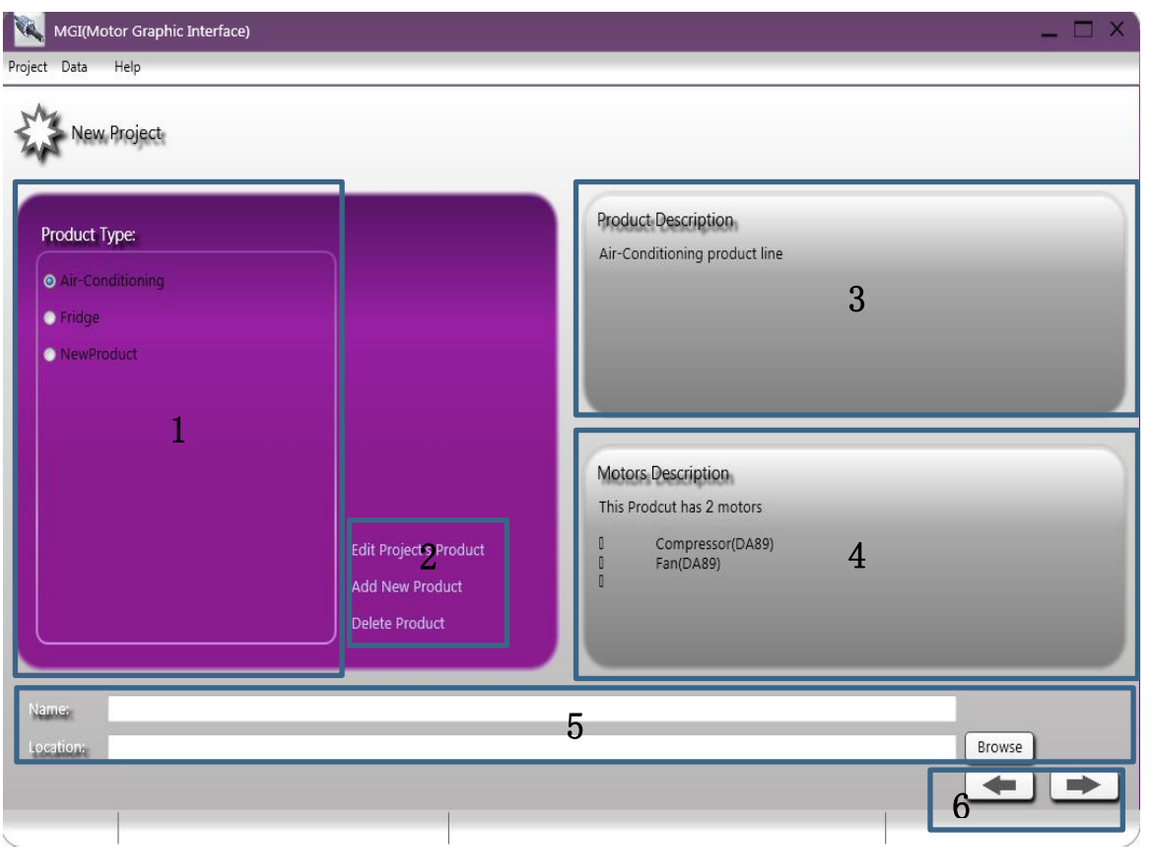


Menu and its items provide interface to use features of this software, such as open project, close project and so on.

3.2 New/Edit Project Page

3.2.1 New Project

Figure 3-3: New Project



At Home page click “New Project” or click menu item “New Project” in Project menu. The “New Project” page will be shown (If click menu item “Edit Project”).

This page provides interface for users to select product type, edit product setting or add a new product type.

Table 3-2: New Project Page Components

Number	Name	Functions
1	Product Type	Select project's product that you want to create.
2	Product Edit	1. Edit Project's Product: Switch to "Edit/Add Product Page" to edit selected products. 2. Add New Product: Switch to "Edit/Add Product Page" to add a new product type. 3. Delete Product: The Product created by "Add New Product".
3	Product Description	Description of current selected products.
4	Motors Description	Motors description of current selected products.
5	Project Path and Name	Click "Browse" button to choose a directory to save project file. Input project name in "Name" text box.
6	Navigation Button	1. Back button: Go back to "Home Page" 2. Previous button: Create project then forward to "Work Page".

3.2.2 Edit Project

At the work page, click "Edit Project" to display the "New Project Page".

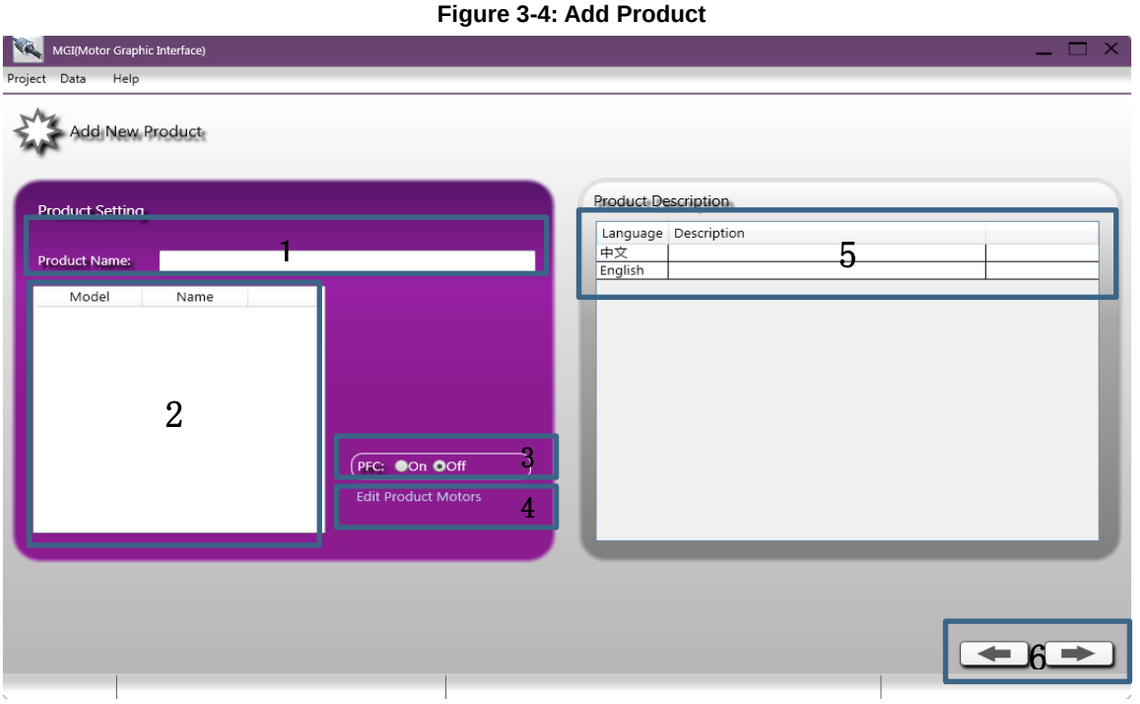
1. The Page Icon and title are changed.
2. The Project Location cannot be changed.

At Edit Project page, current project type and name can be changed.

3.3 Add/Edit Product Page

3.3.1 Add Product

Figure 3-4: Add Product



At the "New/Edit Project" page, click "Add New Product" and the "Add New Product" page will be displayed (Similarly, if you click "Edit Project's Product", the "Edit Product" page will be displayed).

Click “Add Product” to set new product name and the description. Then, click “Edit Product Motors” to display the “Edit Product Motors” page.

Table 3-3: Product Page Components

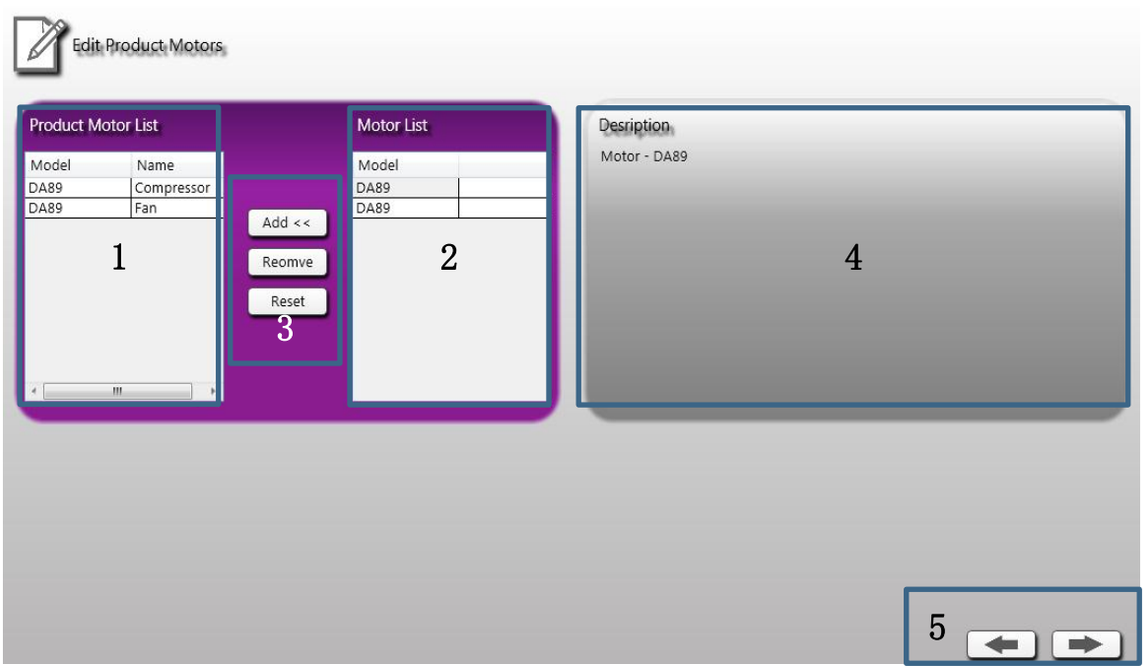
Number	Name	Functions
1	Product Name	Input product name.
2	Product Motor List	This place shows product motors.
3	PFC Enable	Enable or disable PFC
4	Edit Product Motors	Click “Edit Product Motors” to go to “Edit Product Motors” page and then the product’s motors can be added or removed.
5	Product Description	Get input product description in English and Chinese.
6	Navigation Button	Back button: go to the next step. Previous button: back to the previous step

3.3.2 Edit Product

Click “Edit Project’s Product” to display the Product page of the edit mode, which is different from the Add New Product mode, where the product name cannot be edited.

3.4 Edit Product Motors

Figure 3-5: Edit Product Motors



Click “Edit Product Motors” at page “Add/Edit Product” to display the “Edit Product Motors” page. At this page, product’s motors can be added or removed.

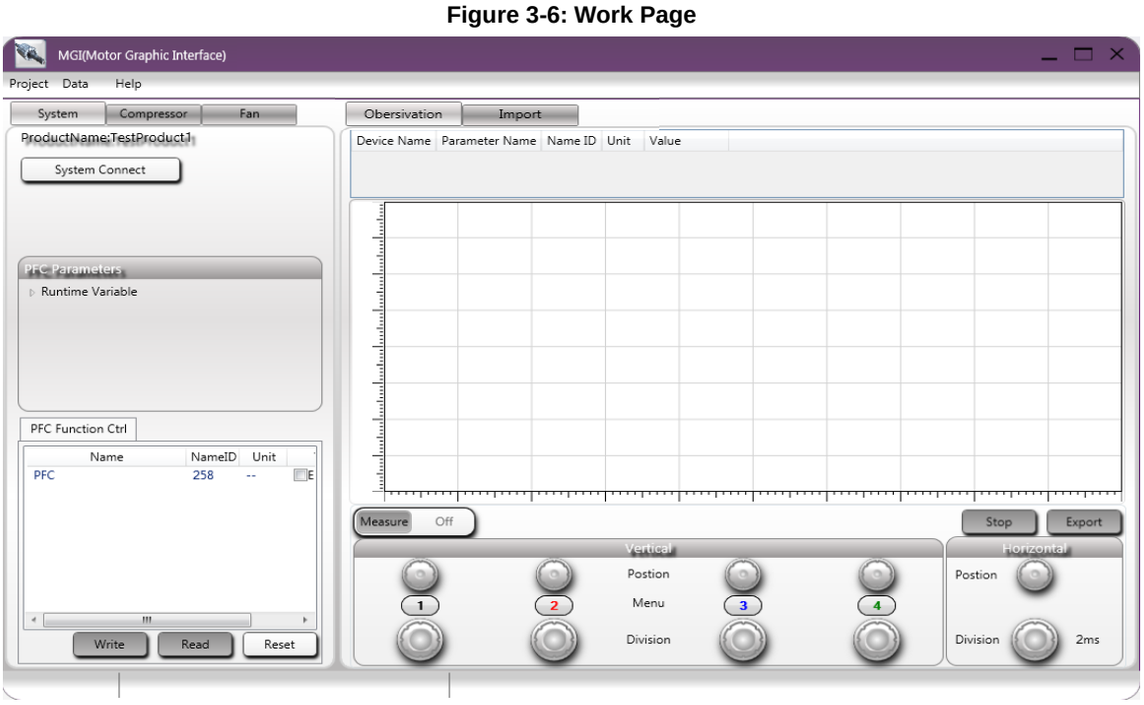
Table 3-4: Product Motors Page Components

Number	Name	Functions
1	Product Motor List	List all motors that current product has. Motors' name can be edited, but the model can't.
2	Motor List	List all motors that can be added to current product.
3	Operation	1. Add the motor selected from the Motor list to Product 2. Remove: Remove selected motor, in the Product Motor List, from product. 3. Reset: Reset product motors
4	Motor Description	Description of selected motors.
5	Navigation Button	Back button: Back to "Add/Edit Product" page without adding or changing the product type. Next button: Go to "Add/Edit Product" page to add or change the product type.

3.5 Work Page

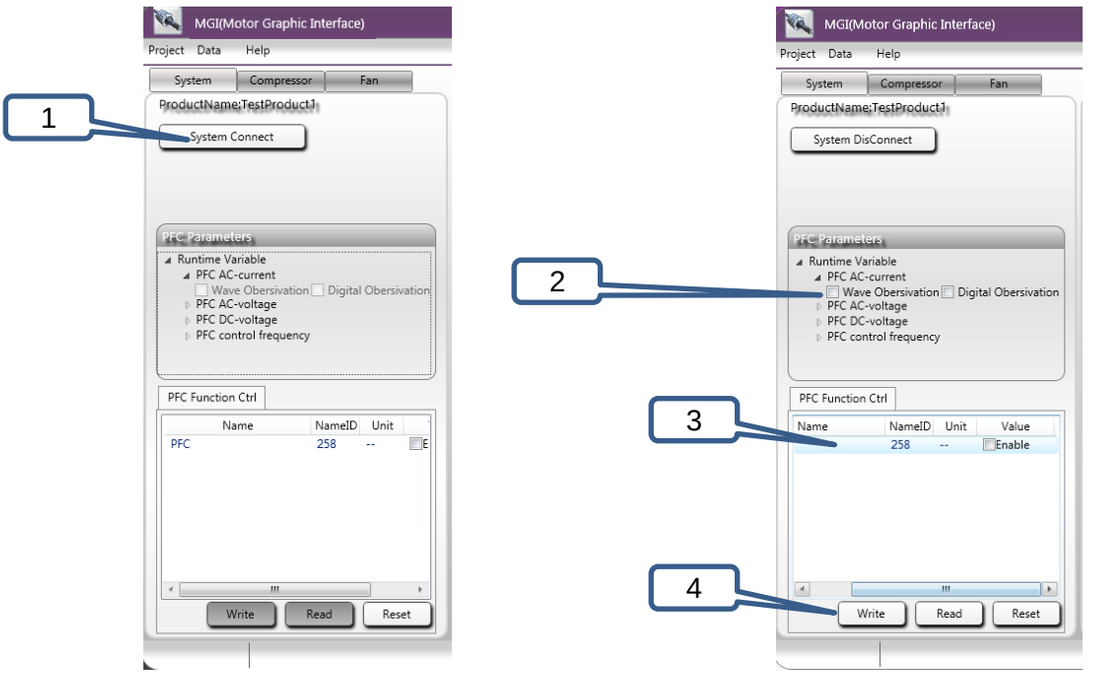
The figure below shows the work page. It has four tabs. System, Motors (one system can has some motors), Observation and Comparison.

Figure 3-6: Work Page



3.5.1 System Tab

Figure 3-7: System Tab



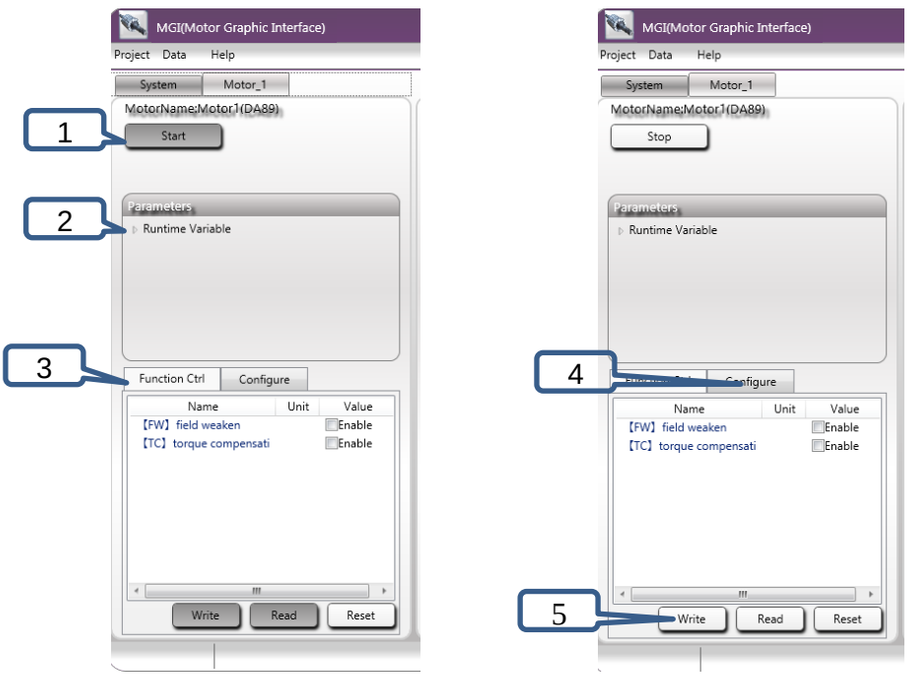
System Tab includes system connection control, PFC parameter observation, and PFC function control. When the system is connected after you click the “System Connect” button, the PFC parameters and Function control area can be operated.

Table 3-5: System Tab Components

Number	Name	Functions
1	System Connect/Disconnect	Connect or disconnect FW (and HW) with this software. If the Parameter observation checkbox is connected, the Function Control check box and Write/Read button are enabled.
2	Parameters Observation List	Parameters in the list provide wave or digital observation. Wave observation can support four lines in whole system; digital observation can support ten parameters.
3	Function Control	Enable or disable functions Now only one function is available – PFC enable/disable Before PFC is enabled, in the region 3 shown in above figure, please confirm: 1. PFC function in FW is enabled. The PFC_EN in HardwareConfig.h is defined as 1. 2. PFC mode in FW is correctly set. The Pfc_u8StartMode in HardwareConfig.c should be set to 1. 3. Please confirm the motor is already in running state, otherwise PFC will not work even if you send a PFC-enable command to FW from SW.
4	Operation Buttons	1. Write: Send parameters’ value to FW. 2. Read: Get parameters’ value from FW. 3. Reset: Reset parameters’ value to default but don’t send to FW

3.5.2 Motor Tab

Figure 3-8: Motor Tab



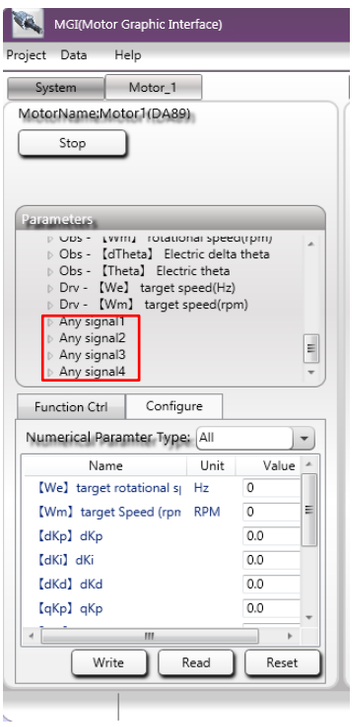
This tab provides motor start/stop control, motor parameters observation, function control and parameter setting.

Table 3-6: Motor Tab Components

Number	Name	Functions
1	Motor Start/Stop Button	Set motor start/stop command to FW for controlling motor start or stop
2	Parameters Observation List	Parameters in the list provide wave and digital observation. Wave observation can support four lines in whole system; digital observation can support ten parameters. In this window, the signals are divided into 3 types: 1. observed signal It describes the observed values in the control system, including current, voltage, back-EMF, speed, angle, etc. 2. driving signal It describes the driving values in the control system, including current, voltage, speed, etc. 3. any signal It offers a signal interface to show any signals that users want to observe. Details will be provided later.
3	Function Control	Enable or disable functions Two functions are available: 1. Field weakening 2. Torque compensation
4	Configure	Setting motor parameters The following parameters are available: 1. Target speed (electrical speed, unit is Hz) 2. Target speed (mechanical speed, unit is RPM) 3. Kp of d-axis close loop 4. Ki of d-axis close loop 5. Kd of d-axis close loop 6. Kp of q-axis close loop 7. Ki of q-axis close loop 8. Kd of q-axis close loop 9. Kp of motor rotational speed close loop 10. Ki of motor rotational speed loop 11. Kd of motor rotational speed loop
5	Operation Buttons	1. Write: Send current tab's (Function Control or Configure) parameters' value to FW. 2. Read: Get current tab's (Function Control or Configure) parameters' value from FW. 3. Reset: Reset current tab's (Function Control or Configure) parameters' value to default but don't send to FW

In the parameter observation list, there exist 4 special signals shown in the red box in below figure, which are used for customers to display any signals.

Figure 3-9: Signals Defined by Users



Let's see how 'any signals' work. In FW, the following definitions are designed in UartGUI.h.

Figure 3-10: Definition of User Interface for 'Any signals'

```
#if(EN_GUI_MOTOR_1 && EN_ANY_SIG_INTERFACE)
    #define ASSIGN_MOTOR1_ANY_SIG1() \
            UartGUI_i32Q15_AnySigData_M1[0] = FALSE;
    #define ASSIGN_MOTOR1_ANY_SIG2() \
            UartGUI_i32Q15_AnySigData_M1[1] = FALSE;
    #define ASSIGN_MOTOR1_ANY_SIG3() \
            UartGUI_i32Q15_AnySigData_M1[2] = FALSE;
    #define ASSIGN_MOTOR1_ANY_SIG4() \
            UartGUI_i32Q15_AnySigData_M1[3] = FALSE;
#endif
```

In the array of UartGUI_i32Q15_AnySigData_M1[4], each array element specifies the data of a 'any signal'. For example,

UartGUI_i32Q15_AnySigData_M1[0] collects the data for 'any signal 1',
UartGUI_i32Q15_AnySigData_M1[1] collects the data for 'any signal 2',
UartGUI_i32Q15_AnySigData_M1[2] collects the data for 'any signal 3'
UartGUI_i32Q15_AnySigData_M1[3] collects the data for 'any signal 4'

That is to say, if we put some data into UartGUI_i32Q15_AnySigData_M1[0], these data value can be displayed in waveform window or value window.

So to apply the ‘any signal’, we should do the following steps:

1. Find a global variable in which the signal value you want to observe is stored. For example, we want to see the variation of the motor rotational speed. We know there is a global variable named CompCtrl_stcPll.i32Q8_EstimWmHzf, which logs the real-time motor rotational speed.
2. Transfer the global variable value to the array element in UartGUI_i32Q15_AnySigData_M1[]

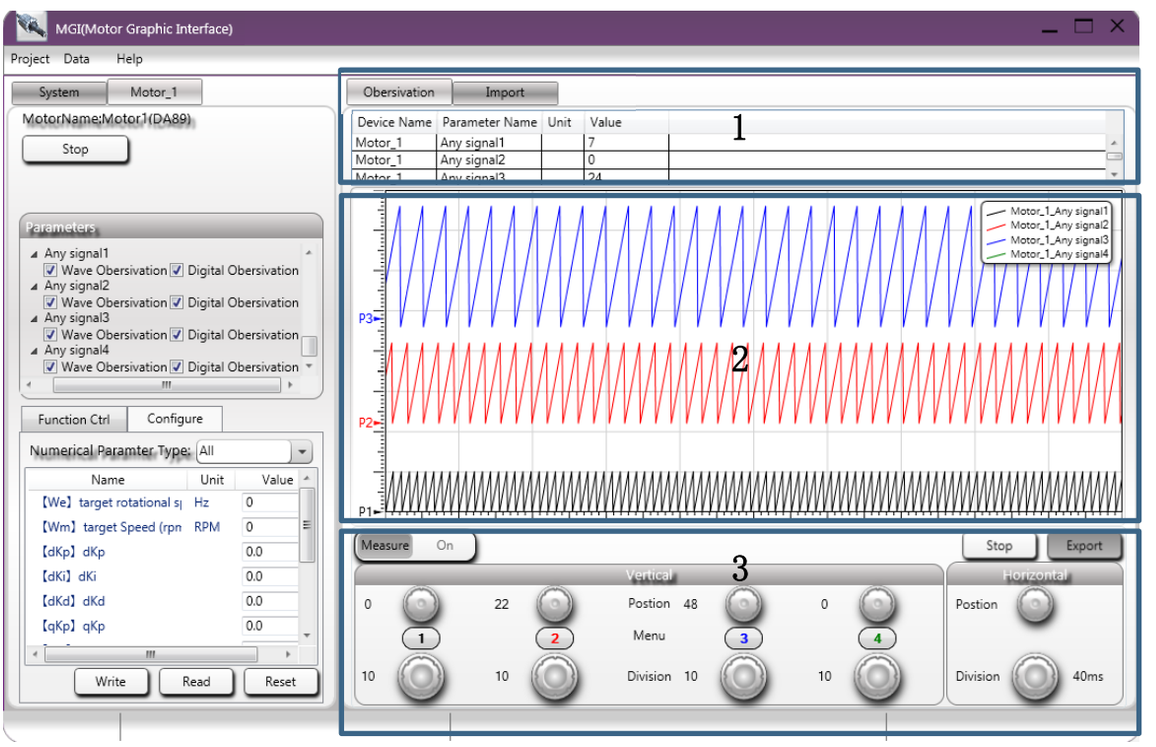
For example, we want to show the waveform of motor rotational speed. We need simply changing the definition as follows.

Figure 3-11: Assign Signal to the ‘Any signal 1’

```
#if(EN_GUI_MOTOR_1 && EN_ANY_SIG_INTERFACE)
    #define ASSIGN_MOTOR1_ANY_SIG1() \
            UartGUI_i32Q15_AnySigData_M1[0] = CompCtrl_stcRunPar.i32TargetSpeedRpm;
    #define ASSIGN_MOTOR1_ANY_SIG2() \
            UartGUI_i32Q15_AnySigData_M1[1] = FALSE;
    #define ASSIGN_MOTOR1_ANY_SIG3() \
            UartGUI_i32Q15_AnySigData_M1[2] = FALSE;
    #define ASSIGN_MOTOR1_ANY_SIG4() \
            UartGUI_i32Q15_AnySigData_M1[3] = FALSE;
#endif
```

3.5.3 Observation Tab

Figure 3-12: Observation Tab



Note: when the frequency of Data Recorder is 500Hz, select 4 channels, the system work normal.

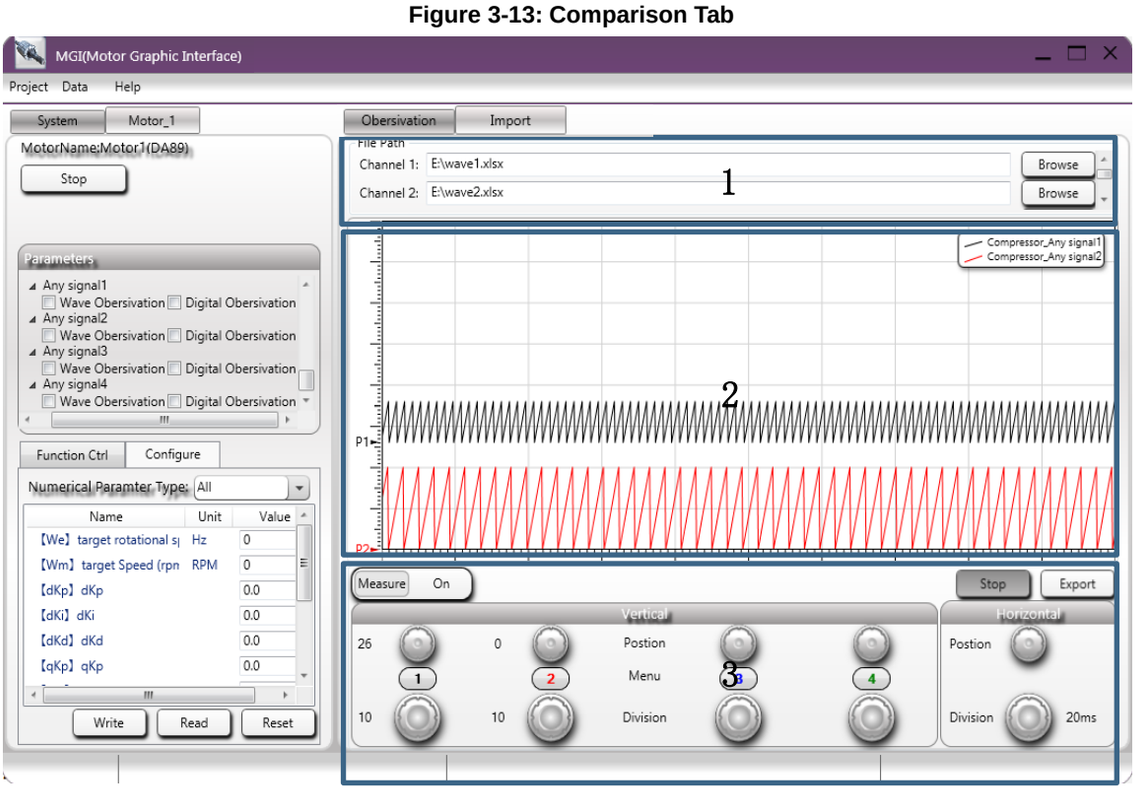
Observation Tab can display observed parameter in system tab or motor tab. The formats for displaying parameter are wave and digital. 4 waves and 10 digitals are supported. Use panel to zoom, move, measure and export wave data.

Table 3-7: Observation Tab Components

Number	Name	Functions
1	Digital	Observe parameters in digital way. 10 parameters are supported.
2	Wave	Observe parameters in wave way. 4 parameters are supported.
3	Panel	1. Measure Button: When any wave exists and current state is stop, measure function is enabled. It can measure value in x and y axis. 2. Start/Stop Button: Start or stop displaying the wave. 3. Export: When any wave exists and current state is stop, the export button is enabled, it can export each channel's data into excel file or text file, and it also can get a screen shot of wave area. 4. Vertical Position Button: Each wave has its own vertical position setting button, use these buttons to set relative button's vertical position. 5. Show/Hide Button: Show or hide relative button. 6. Vertical Division Button: Change the division of relative wave in y axis. 7. Horizontal Position Button: This button can be used in stop state and waves are not completely displayed in x axis. 8. Horizontal Division Button: Change the x axis division for all waves.

3.5.4 Import Tab

Figure 3-13: Comparison Tab



At this page, waves exported before can be imported.

Table 3-8: Comparison Tab Components

Number	Name	Functions
1	Wave Files	Observe parameters in digital way. 10 parameters are supported.
2	Wave	Same as chapter 3.5.3 Observation Tab.
3	Panel	Same as chapter 3.5.3 Observation Tab, except the stop button is disabled.

4. Use Example

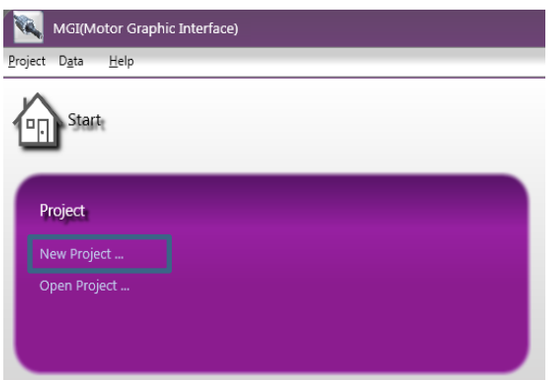
4.1 Driver Installation

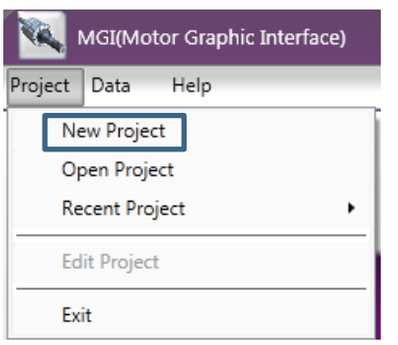
FTDI device driver has two ways to install. First, connect FTDI device to USB port of PC/Laptop and ensure PC/Laptop is connected to internet, driver of FTDI device will be automatically installed by Windows; second, get the FTDI device package then find the driver installation guide in the package, perform steps described in the driver installation guide. For more information of FTDI driver please refer to its website *www.ftdi.com*.

4.2 New Project

1. Click “New Project” at Home Page or select “New Project” Item from Project Menu.

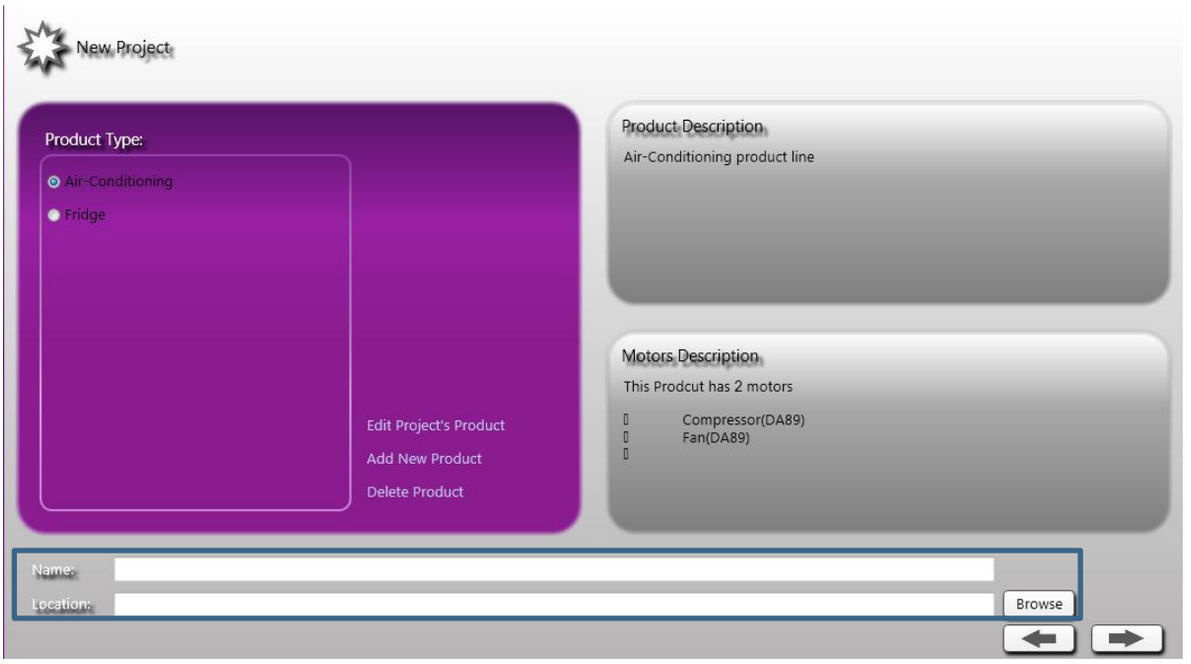
Figure 4-1: New Project Step1





2. At New Project Page, select a project product type from product list or add a new product by clicking “Add New Product”. Click “Edit Project’s Product” to modify current selected product type’s setting for current new project. For more details about adding product or editing project’s product, please refer to *Chapter*4.4 Edit/Add Product. After selecting a product type, input a project name and then input the location or use “Browse” button to select the location for saving project file path. At last click Next Arrow button to create a new project.

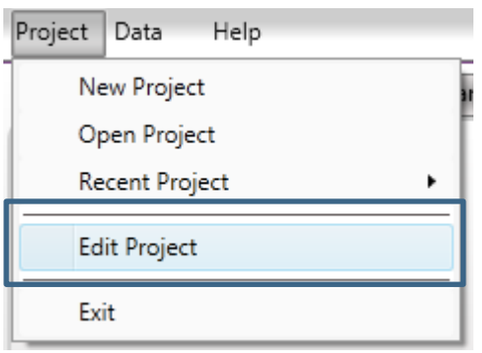
Figure 4-2: New Project Step2



4.3 Edit Project

1. If you need to modify an opened project, select “Edit Project” from the Project Menu.

Figure 4-3: Edit Project Step1



2. Product Type and Name can be modified if needed. At last, click Next Arrow button to save the change of project and open it. For more information about “Edit Project’s Product” and “Add New Product”, please refer to Chapter 4.4 *Edit/Add Product*.

Figure 4-4: Edit Project Step2

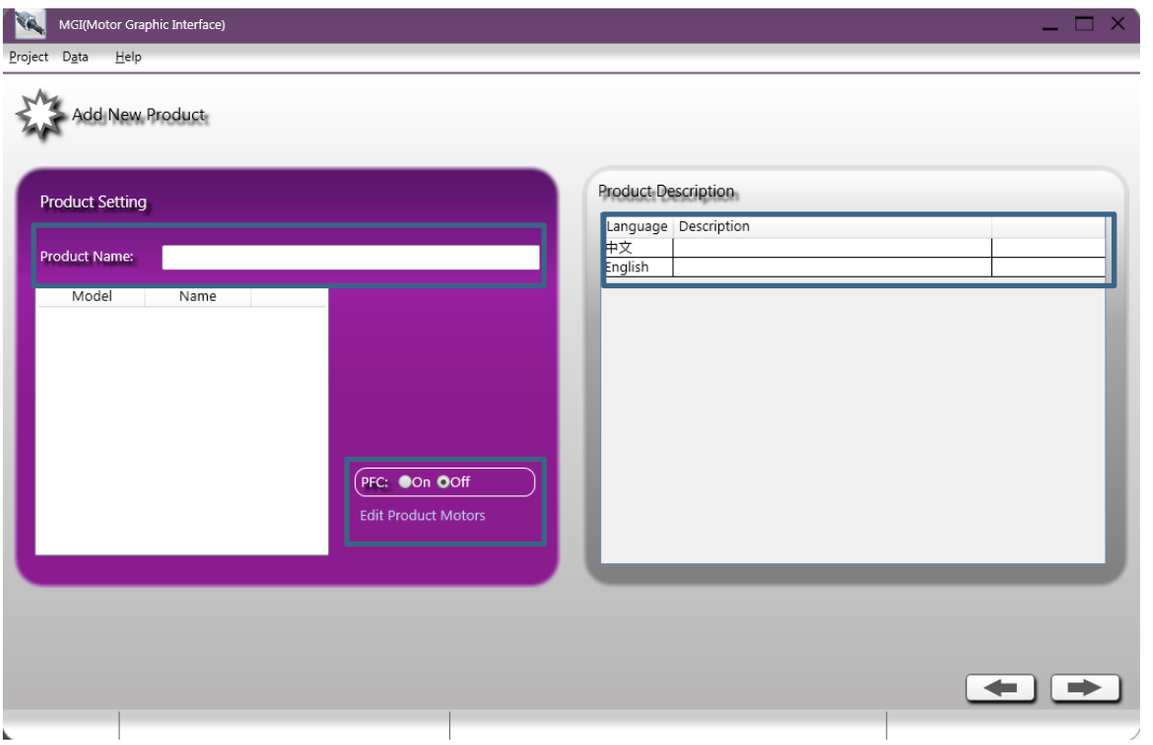
4.4 Edit/Add Product

To create a new project or edit current project, existing product types may not meet the requirements. In this case, “Add New Product” and “Edit Project’s Product” are needed.

4.4.1 Add Product

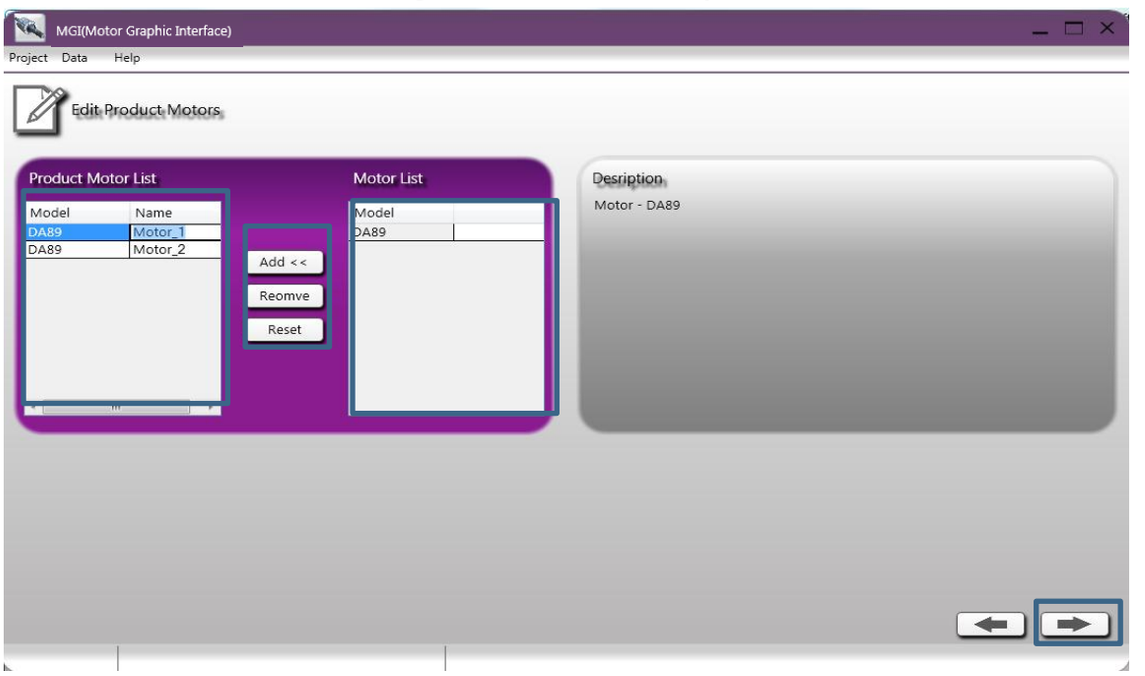
1. Click “Add New Product” at New/Edit Project page to display the “Add Product” Page.
2. Input the Product Name, and then input product description. Now the product doesn’t have any motors. Click “Edit Product Motors” to go to the next step.

Figure 4-5: Add Product Step2



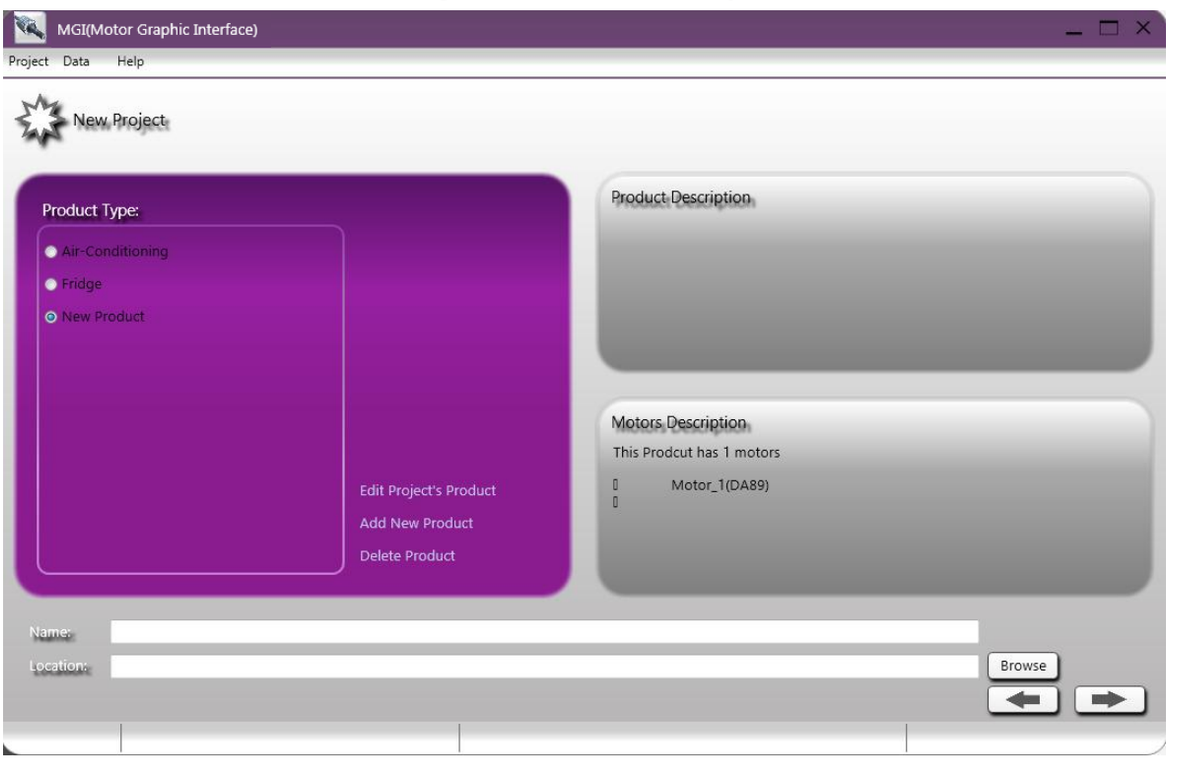
3. Select a motor from the Motor List and then click “Add <<” button to add the motor to the product. Motor Name in Product Motor List can be edited. At last, click “Next Arrow” button to finish the procedure for adding new product.

Figure 4-6: Add Product Step3



4. Now the new product type is listed in the Product Type Area, which can be used to create new project.

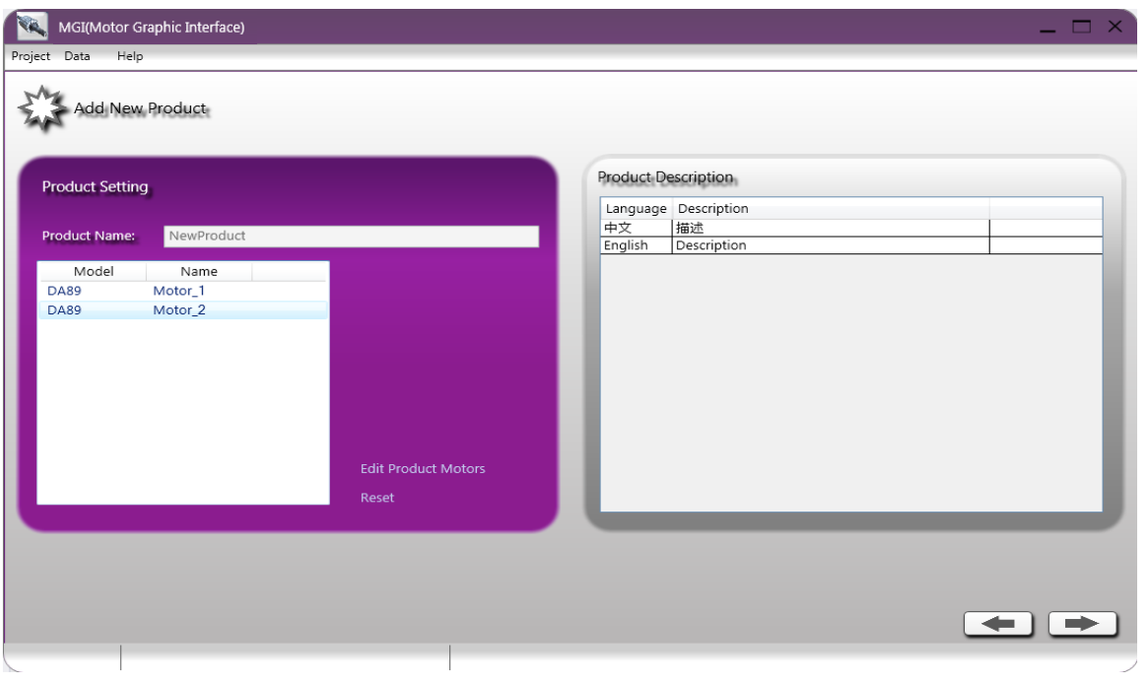
Figure 4-7: Add Product Step4



4.4.2 Edit Product

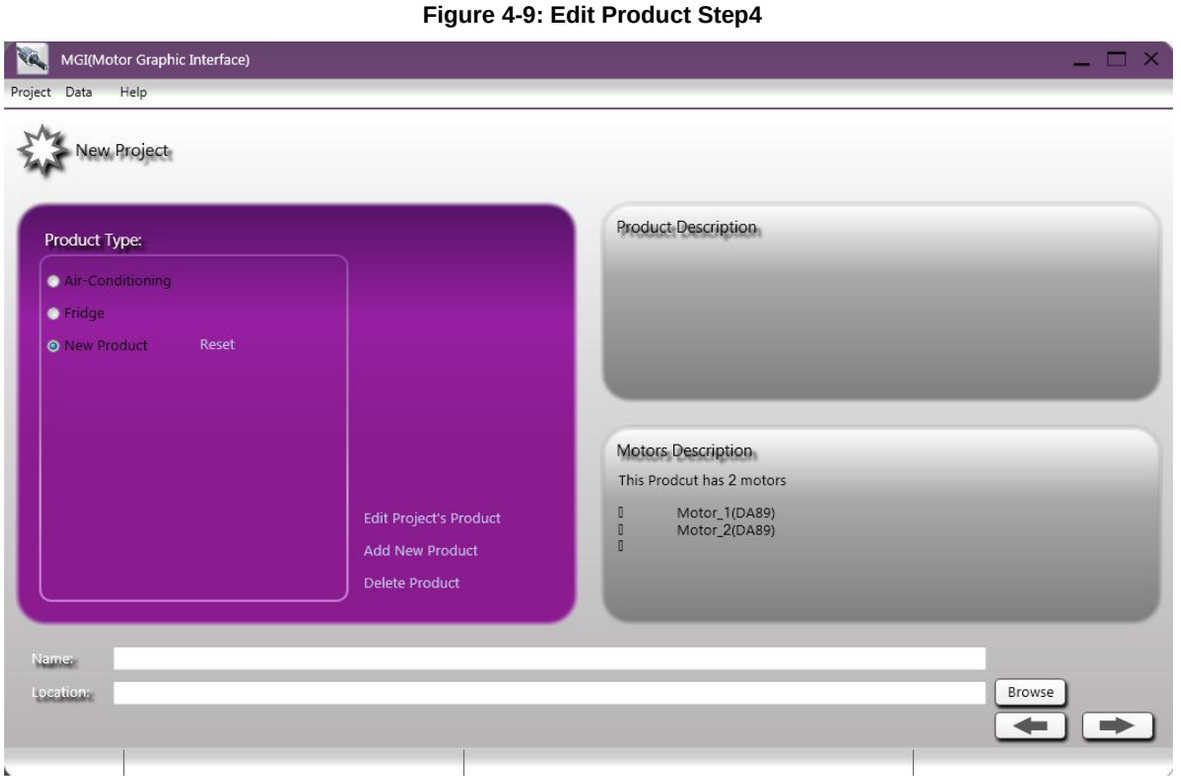
1. Click “Edit Project’s Product” at New/Edit Project page to display the “Add Product” Page.
2. Product Description can be edited at this page. Click “Edit Product Motors” to add or remove motors of product.

Figure 4-8: Edit Product Step2



3. Edit product motors. For this step, please refer to *Chapter 4.4.1 Add Product*.
4. The Edit Product function can be used to create a new project. If you click the “Reset” after the edit product name, the edit product will be reset.

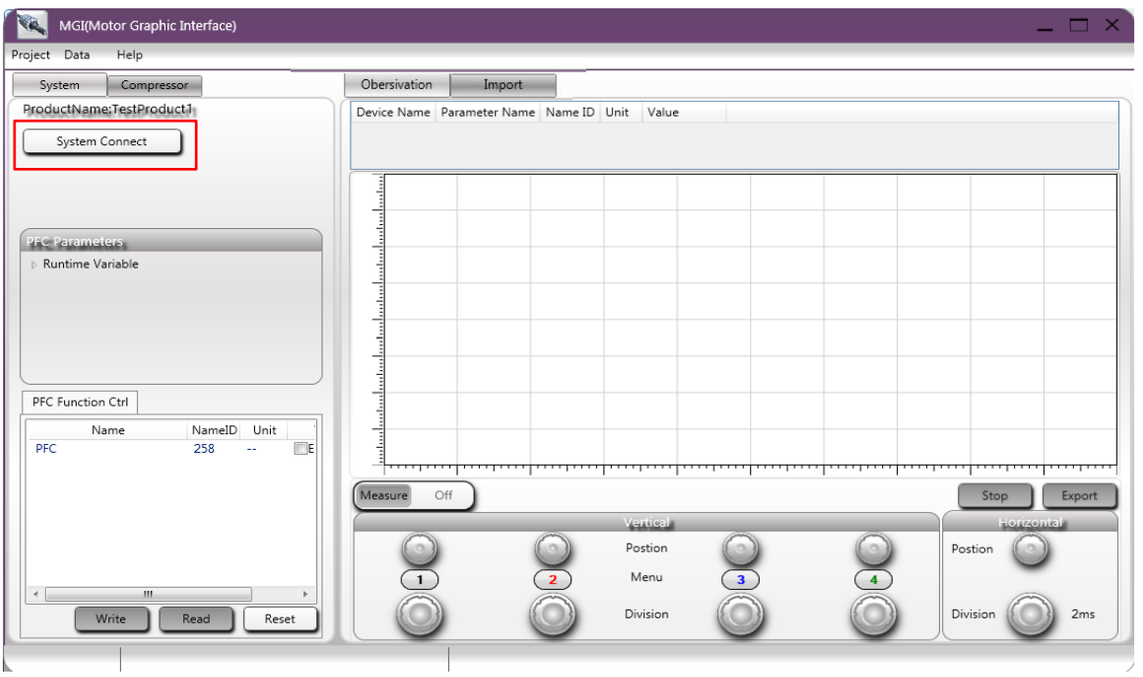
Figure 4-9: Edit Product Step4



4.5 Connect/Read/Write

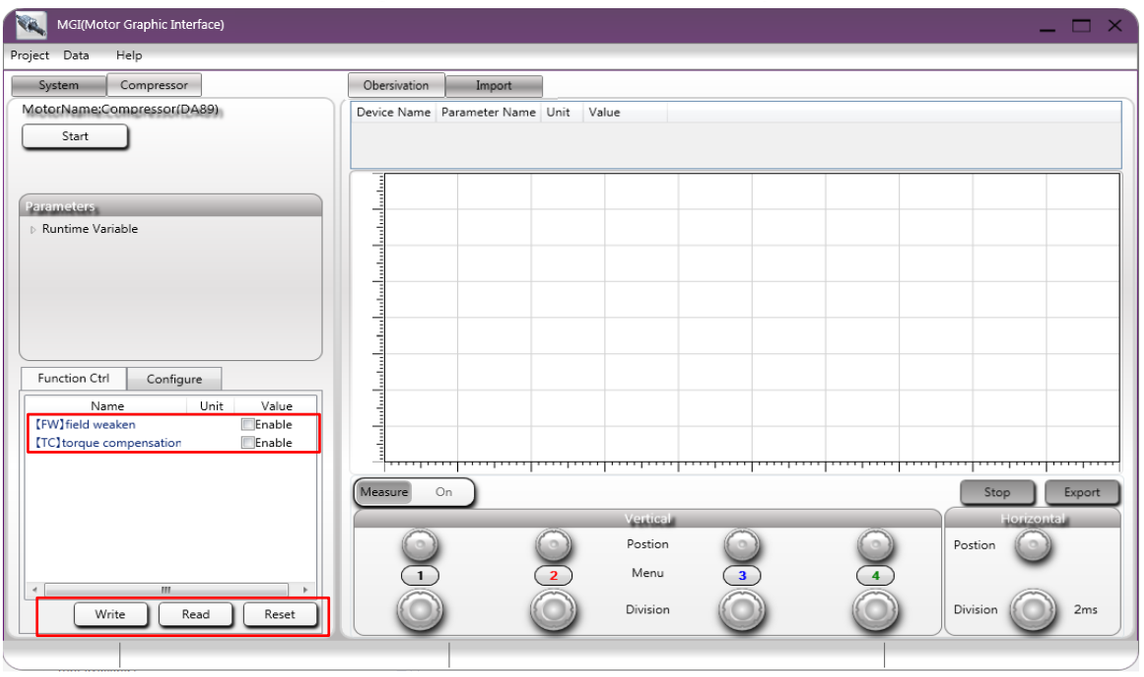
1. Open an existing project or create a new project.
2. Click “System Connect” button to connect SW to FW.

Figure 4-10: Connect/Read/Write Step2



3. Change the Function Control or configure parameter values and then press “Write” button. Changed values will be sent to FW. Click “Read” button to read parameters’ value from FW. Click “Reset” button to set values to the default value.

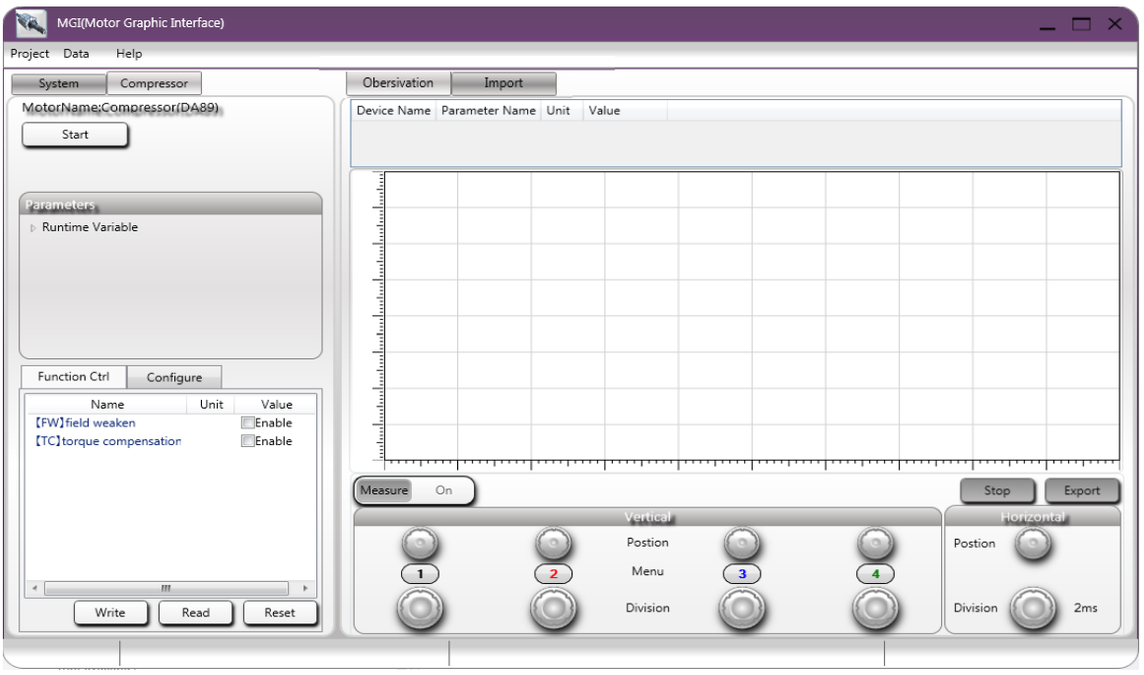
Figure 4-11: Connect/Read/Write Step3



4.6 Observation

1. At first open an existing project or create a new project.
2. Click “System Connect” button at Work Page.
3. Choose the parameter that you want to observe. There are two observation ways: digital and wave. Use the panel to tune the wave display, measure wave or export wave data.

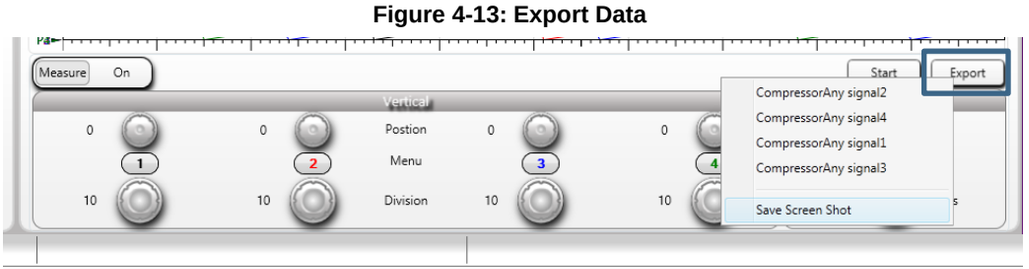
Figure 4-12: Observation Step3



4.7 Export Data and Import Data

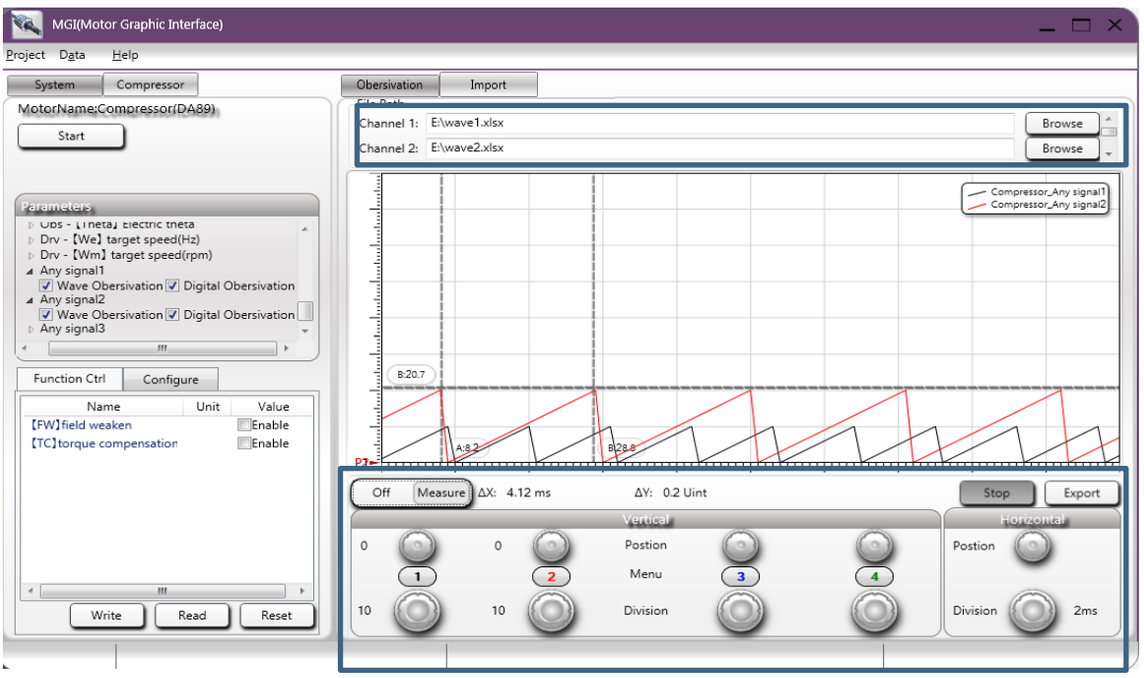
1. Click “Export” to select signal to save, click “Export->Save Screen Shot” to capture the wave picture

Figure 4-13: Export Data



1. Open a project or create a new project.
2. Choose the Comparison Tab.
3. Open an existing wave data file by clicking the “Browse” button. After waves are loaded, use panel to tune wave display, measure or export wave screenshot.

Figure 4-14: Import Data



5. Customer Control Interface in FW

Some definitions in UartGUI.h can be modified by users to control the GUI function in FW.

5.1 UART Selection

Table 5-1: UART Selection

Macro definition	Description
GUI_FMx	Choose the MFS to be assigned for the GUI. For the demo, MFS6 is selected.

5.2 Enable/Disable Switch

Table 5-2: Enable/Disable Switch

Macro definition	Description
EN_UART_GUI	TRUE – enable FW to run GUI function FALSE – disable FW from running GUI function If it is disabled, the GUI function in FW will not run.
EN_GUI_SYS	TRUE – enable FW to run GUI function relative to system control/observation FALSE – disable FW from running GUI function The system function, by now, only includes PFC. So if it is true, the PFC can be controlled and observed. Otherwise, the PFC operation in GUI is not available.
EN_GUI_MOTOR_1	TRUE – enable FW to run GUI function relative to motor1 control/observation FALSE – disable FW from running GUI function relative to motor1 control/observation
EN_GUI_MOTOR_2	It is not available, that is only one motor is supported by now. So please set it to FALSE.
EN_ANY_SIG_INTERFACE	TRUE – enable FW to run GUI function relative to any signal observation. FALSE – disable FW from running GUI function relative to any signal observation.

5.3 Dotting Frequency Control for Waveform and Value Display

Table 5-3: Dotting Frequency Control

Macro definition	Description
M2OC_PACK_TIMING_CNT_SYS	It acts as dotting frequency control for the system function (PFC). For example, the PFC control runs at every FRT zero-match interrupt. So the PFC control frequency is the same as the FRT zero-match interrupt frequency. Let's define a short name for the interrupt frequency as ZM-freq, and also the ZM-period is the brief name for FRT zero-match interrupt period. If the ZM-freq is 5K, the ZM-period should be 200us. If M2OC_PACK_TIMING_CNT_SYS = 1, it means a

	waveform dot will be created in every ZM-period. If M2OC_PACK_TIMING_CNT_SYS = 2, it means a waveform dot will be created in every 2 ZM-periods. … If M2OC_PACK_TIMING_CNT_SYS = k, it means a waveform dot will be created in every k ZM-periods. When MCU frequency is low, GUI waveform display probably cannot get enough time to run normally because the PFC/motor control function is prior to GUI and occupy so much MCU running time. In this condition, we have to increase the k to a bigger value to assure GUI waveform display can run normally. Please note that the bigger k causes the lower waveform sample rate. The waveform may be distorted if the k is too big. In other hand, the k value cannot be too small; otherwise the data quantity will exceed the ability of MGI drawing. Here has a recommend value to determine k value: the curve dotting frequency do not exceed 500Hz.
M2OV_PACK_TIMING_CNT_SYS	Please refer to M2OC_PACK_TIMING_CNT_SYS The difference is the counter is for value display. For value display variables, the dotting frequency is recommend to 1 time per second.
M2OC_PACK_TIMING_CNT_M1	Please refer to M2OC_PACK_TIMING_CNT_SYS
M2OV_PACK_TIMING_CNT_M1	Please refer to M2OV_PACK_TIMING_CNT_SYS
M2OC_PACK_TIMING_CNT_M2	Please refer to M2OC_PACK_TIMING_CNT_SYS
M2OV_PACK_TIMING_CNT_M2	M2OV_PACK_TIMING_CNT_SYS

5.4 Observation of User-Defined Signals

Table 5-4: Observation of User-Defined Signals

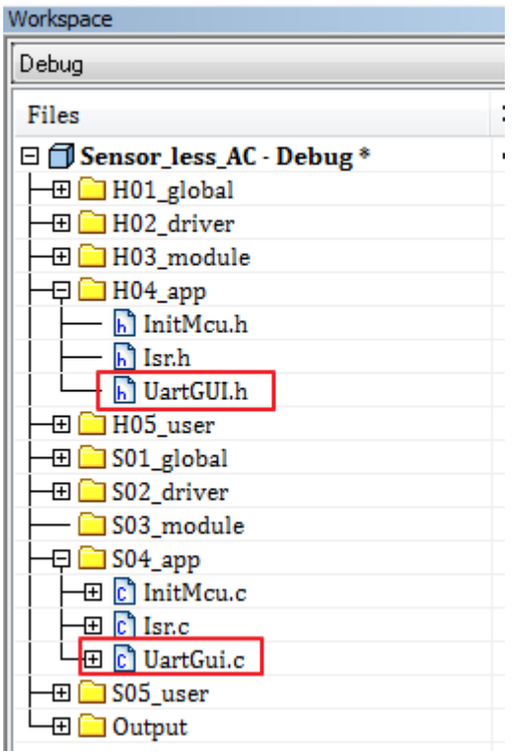
Macro definition	Description
UartGUI_i32Q15_AnySigData_SYS[0]	It is reserved and not used by now.
UartGUI_i32Q15_AnySigData_SYS[1]	
UartGUI_i32Q15_AnySigData_SYS[2]	
UartGUI_i32Q15_AnySigData_SYS[3]	
UartGUI_i32Q15_AnySigData_M1[0]	Please refer to ‘3.5.2 Motor Tab’
UartGUI_i32Q15_AnySigData_M1[1]	
UartGUI_i32Q15_AnySigData_M1[2]	
UartGUI_i32Q15_AnySigData_M1[3]	
UartGUI_i32Q15_AnySigData_M2[0]	It is reserved and not used by now.
UartGUI_i32Q15_AnySigData_M2[1]	
UartGUI_i32Q15_AnySigData_M2[2]	
UartGUI_i32Q15_AnySigData_M2[3]	

6. Transplanting MGI to Your Application

6.1 Necessary Files

There are 2 files are necessary to insert into your application to support MGI: UartGui.c and UartGui.h.
The 2 files can be copied from the demo project and then put them to any place in your application folder.

Figure 6-1: Necessary Files to Support MGI



6.2 Configure UART to Meet Your Hardware

To normally run the MGI, it is necessary to configure UART according to your hardware design.
In the demo project, an initialization function named InitMcu_Uart0() is to configure the UART according to the demo hardware (shown as below). Because the initialization function calls some sub-functions which are the component belong to the firmware architecture in driver layer, maybe it is not suitable to combine it with your code. If so, please find another uart-initialization function in other application to meet your requirement.

Figure 6-2: MCU UART Initialization

```
    stc_uart0_cfg_t stUI_Uart0;

#if(EN_UART_GUI)
    stUI_Uart0.PortPin_Rx.Port = PORT3; // Rx port and pin
    stUI_Uart0.PortPin_Rx.Pin = PIN3;
    stUI_Uart0.PortPin_Tx.Port = PORT3; // Tx port and pin
    stUI_Uart0.PortPin_Tx.Pin = PIN2;
    stUI_Uart0.Baudrate = 2000000; // baudrate
    stUI_Uart0.Mode.All = 0;
    stUI_Uart0.Mode.bits.ParityEn = TRUE; // enable parity check
    stUI_Uart0.Mode.bits.ParitySel = ODD_NUMBER_PARITY; // odd check
    stUI_Uart0.Mode.bits.DataLen = 8; // data bit len = 8-bits
    stUI_Uart0.Mode.bits.StopBitLen = 1; // stop bit len = 1-bit
    stUI_Uart0.Mode.bits.BitInvertEn = FALSE; // do not inverse bit
    stUI_Uart0.Mode.bits.Direction = LSB_FIRST; // data transfer dir = LSB
    stUI_Uart0.Mode.bits.FlowCtrlEn = FALSE; // flow control is disabled
    stUI_Uart0.Mode.bits.DataOutputEn = TRUE; // enable data output
    stUI_Uart0.EnRxINT = TRUE; // enable Rx interrupt
    stUI_Uart0.EnTxINT = FALSE; // disable Tx interrupt

    stUI_Uart0.EnRxFIFO = FALSE; // do not use Rx FIFO
    stUI_Uart0.EnTxFIFO = FALSE; // do not use Tx FIFO
    stUI_Uart0.RxFifoLen = 8; // only Rx FIFO length can be set

    if (!SetUart0(&stUI_Uart0))
    {
        return FALSE;
    }
#endif
```

6.3 Add Code to Your Application

- Step 1: configuring UART communication variables
  The function name is UartGUI_Init().
  It should be put before the main-while-loop.

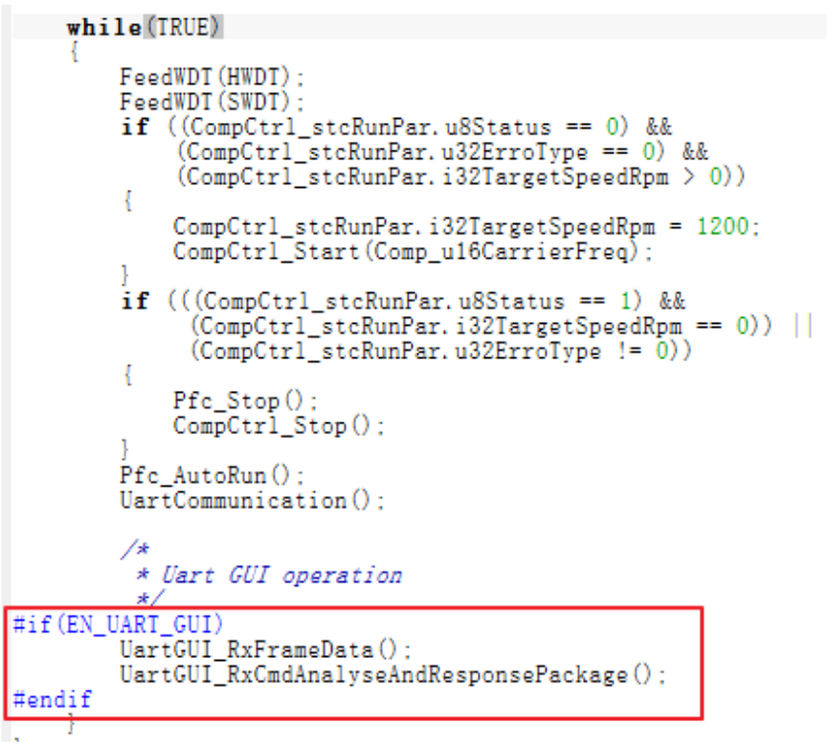
Figure 6-3: Configuring UART Communication Variables

```
void main(void)
{
    InitMcu_Clock();
    InitMcu_Nvic();
    InitMcu_Gpio();
    InitMcu_Basetimer();
    InitMcu_Uart0();

    InitMcu_Adc(Comp_u16CarrierFreq);
    CompCtrl_Init(Comp_u16CarrierFreq);
    Pfc_Init();
    InitMcu_CompTimerRun();
    Pfc_TimerRun(300);
    InitMcu_Wdg();
    Pfc_Stop();

#if(EN_UART_GUI)
    UartGUI_Init();
#endif

    while(TRUE)
    {
        FeedWDT(HWDT);
        FeedWDT(SWDT);
```

- Step 2: adding the functions of receiving data and analyzing data
  The functions name are: UartGUI_RxFrameData() and UartGUI_RxCmdAnalyseAndResponsePackage().

They should be put in the main-while-loop.
The data receiving function is designed to save all bytes received from UART.
The data analysis function is designed to recognize what commands have been given by PC.

Figure 6-4: Adding the Functions of Receiving Data and Analyzing Data

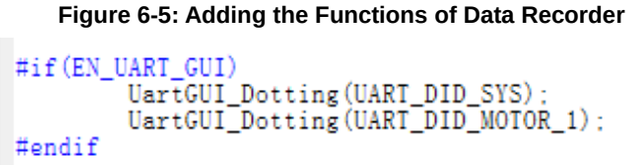


```
    while(TRUE)
    {
        FeedWDT(HWDT);
        FeedWDT(SWDT);
        if ((CompCtrl_stcRunPar.u8Status == 0) &&
            (CompCtrl_stcRunPar.u32ErroType == 0) &&
            (CompCtrl_stcRunPar.i32TargetSpeedRpm > 0))
        {
            CompCtrl_stcRunPar.i32TargetSpeedRpm = 1200;
            CompCtrl_Start(Comp_u16CarrierFreq);
        }
        if (((CompCtrl_stcRunPar.u8Status == 1) &&
             (CompCtrl_stcRunPar.i32TargetSpeedRpm == 0)) ||
             (CompCtrl_stcRunPar.u32ErroType != 0))
        {
            Pfc_Stop();
            CompCtrl_Stop();
        }
        Pfc_AutoRun();
        UartCommunication();

        /*
         * Uart GUI operation
         */
#if(EN_UART_GUI)
        UartGUI_RxFrameData();
        UartGUI_RxCmdAnalyseAndResponsePackage();
#endif
    }
```

- Step 3: adding the functions of data recorder

  The functions name are: UartGUI_Dotting(UART_DID_SYS) and UartGUI_Dotting(UART_DID_MOTOR_1). Please do not forget include the parameter in the bracket.
  They should be put in a proper interrupt routine. For example, it can be put in the interrupt of motor control. Also it can be put in the interrupt of motor current ADC sample. Furthermore, you can put it into a regular timer interrupt.
  Data recorder is to regularly record data to be observed. A data dot is recorded when the interrupt runs once. Because the interrupt runs regularly, the data dots are recorded into a data buffer by synchrony.

Figure 6-5: Adding the Functions of Data Recorder

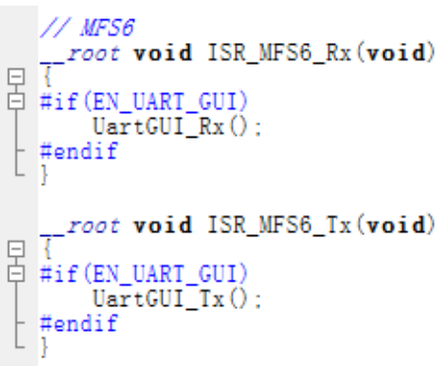


```
#if(EN_UART_GUI)
        UartGUI_Dotting(UART_DID_SYS);
        UartGUI_Dotting(UART_DID_MOTOR_1);
#endif
```

- Step 4: adding the functions of Rx and Tx interrupt

  The functions name are: UartGUI_Rx() and UartGUI_Tx()
  Put them to the proper interrupt entrance. For example, if UART6 is used, add them into MFS6 interrupt entrance shown as follows.

Figure 6-6: Adding the Interrupt Functions



```
// MFS6
__root void ISR_MFS6_Rx(void)
{
#if(EN_UART_GUI)
    UartGUI_Rx();
#endif
}

__root void ISR_MFS6_Tx(void)
{
#if(EN_UART_GUI)
    UartGUI_Tx();
#endif
}
```

- Step 5: configure the interrupt priority

  It is important to assure a correct interrupt priority. Please follow the 2 rules:

  1. Rx interrupt priority should be as high as possible

     In the demo project, Rx interrupt priority is 1, only lower than motor over-current hardware interrupt.
  2. Tx interrupt priority can be at a low priority

     In the demo project, Tx interrupt priority is 15, it is the lowest priority.

7. Troubleshooting

7.1 Connection Error

SW cannot connect to FW by clicking “System Connect” button.

1. Check whether the hardware board is connected to a PC/Laptop and ensure it is powered on.
2. Check whether the FTDI driver is installed.
3. Re-click “System Connect” button, if this does not work. Go to step 4.
4. Re-open SW and re-connect hardware, and then use the SW to connect FW.

7.2 Operation Error

1. Check whether the connection is OK or not.
2. Re-operate, if it also gets failed, re-start the SW and Reset hardware.

7.3 Operation Warning

Some warning will occur when this software is used. The list below shows all warnings.

1. Observation waves are out of range. The whole system can support 4 waves.
2. Observation digital signals are out of range. The whole system can support 10 digital signals.

8. Additional Information

For more Information on Spansion semiconductor products, visit the following websites:
English version address:
http://www.spansion.com/Products/microcontrollers/Pages/default.aspx
Chinese version address:
http://www.spansion.com/CN/Products/microcontrollers/Pages/index.aspx

Please contact your local support team for any technical question
America: Spansion.Solutions@Spansion.com
China: mcu-ticket-cn@spansion.com
Europe: mcu-ticket-de@spansion.com
Japan: mcu-ticket-jp@spansion.com
Other: http://www.spansion.com/Support/SES/Pages/Ask-Spansion.aspx

9. Reference Documents

[1]. AN_104_FTDI_Drivers_Installation_Guide_for_WindowsXP (FT_000093).pdf: FTDI device driver installation guide for Windows XP.

[2]. AN_119_FTDI_Drivers_Installation_Guide_for_Windows7.pdf: FTDI device driver installation guide for Windows 7.

[3]. AN_234_FTDI_Drivers_Installation_Guide_for_Windows_8.pdf: FTDI device driver installation guide for Windows 8.

AN706-00063-1v0-E

Spansion • User Manual

FM3 Family
32-BIT MICROCONTROLLER
MGI User Manual

June 2014 Rev. 1.0

Published: Spansion Inc.
Edited: Embedded System Plat Develop-Em.

Colophon

The products described in this document are designed, developed and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for any use that includes fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for any use where chance of failure is intolerable (i.e., submersible repeater and artificial satellite). Please note that Spansion will not be liable to you and/or any third party for any claims or damages arising in connection with above-mentioned uses of the products. Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions. If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Law of Japan, the US Export Administration Regulations or the applicable laws of any other country, the prior authorization by the respective government entity will be required for export of those products.

Trademarks and Notice

The contents of this document are subject to change without notice. This document may contain information on a Spansion product under development by Spansion. Spansion reserves the right to change or discontinue work on any product without notice. The information in this document is provided as is without warranty or guarantee of any kind as to its accuracy, completeness, operability, fitness for particular purpose, merchantability, non-infringement of third-party rights, or any other warranty, express, implied, or statutory. Spansion assumes no liability for any damages of any kind arising out of the use of the information in this document.

Copyright © 2013 Spansion Inc. All rights reserved. Spansion®, the Spansion logo, MirrorBit®, MirrorBit® Eclipse™, ORNAND™ and combinations thereof, are trademarks and registered trademarks of Spansion LLC in the United States and other countries. Other names used are for informational purposes only and may be trademarks of their respective owners.